VISUAL BASIC – ZÁKLADNÍ PŘÍKAZY

(podle http://www.5starsupport.com/info/vb.htm , 2004)

: Separates commands in a line

Strings

Left Returns the left n characters of a string
```
temp$ = left$ ( teststring$, 4 )
```
Right Returns the right n characters of a string
```
temp$ = right$ ( teststring$, 4 )
```
Trim Removes leading and trailing spaces of a string
```
temp$ = trim$ ( teststring$ )
```
LTrim Removes only the leading spaces of a string
```
temp$ = ltrim$ ( teststring$ )
```
RTrim Removes only the trailing spaces of a string
```
temp$ = rtrim$ ( teststring$ )
```
UCase Makes all characters upper case
```
temp$ = ucase$ ( teststring$ )
```
LCase Makes all characters lower case
```
temp$ = lcase$ ( teststring$ )
```
Mid Returns n characters from a string, starting a any position
```
temp$ = mid$ ( teststring$, 1, 4 )
```
Len Returns the length of a string (how many characters it has)
```
temp$ = len ( teststring$ )
```
LSet Positions a string inside another, flush to the left
```
temp$ = lrset ( teststring$ )
```
RSet Positions a string inside another, flush to the right
```
temp$ = rset$ ( teststring$ )
```
Format Returns a string formatted according to a user-defined format
```
temp$ = format$ ( teststring$, "####.0" )
```
String Vytvoří řetězec požadované délky z prvního písmene řetězce
```
temp$ = String$(4, „A") ...> „AAA"
```
Chr Returns the string representation of a number
```
temp$ = str$ ( 32 )
```
Asc Returns the ASCII code of a single character
```
temp$ = asc ( "A" )
```
Space Returns n spaces
```
temp$ = space$ ( 15 )
```
Instr Determines if one string is found within a second string
```
i = Instr (starthere, string1, string2)
```
InStrRev Determine if one string is found in a second, starting at the end; `i = InStrRev (string1, string2, start)`

StrComp Compares two strings
```
result = StrComp (string1, string2)
```
StrConv Converts the case of a string's characters
```
StrConv (string, vbuppercase)
```
StrReverse Reverses character order in a string
```
StrReverse (string1)
```
Replace Replaces each occurrence of a string
```
Replace (bigstring, searchstring, replacementstring)
```
FormatCurrency Returns a string using a currency format
```
FormatCurrency(var1, 2)
```
FormatDateTime Returns a date or time expression
```
FormatDateTime("3/2/99",vbShortTime)
```
FormatNumber Returns a number formatted according to a variety of options; `FormatNumber(var1, 2)`

FormatPerCent Returns a number formated as a percent
```
FormatPerCent(var1, 2)
```

Operators

/ Normal division
\ Integer division (truncates the answer)
^ Exponentiation operator
* Multiply
+ Plus
- Minus
= Equal
> Greater Than
< Less Than
<> Not Equal
>= Greater than or equal
<= Less than or equal

AND Defines a boolean value that is the AND of two values
```
result = expression1 AND expression2
```
OR Defines a boolean value that is the OR of two values
```
result = expression1 OR expression2
```
XOR Defines a boolean value that is the exclusive OR of two values
```
result = expression1 XOR expression2
```
NOT Defines an opposite boolean value
```
A = NOT B
```
EQV Performs a logical equivalence on two expressions (result is true if both expressions are true)
```
result = expression1 EQV expression2
```
IMP Performs a logical implication on two expressions
```
result = expression1 IMP expression2
```
IS Determines if 2 variables reference the same object
```
result = object1 IS object2
```
LIKE Determines if one string matches a pattern
```
result = string LIKE pattern
```
MOD Returns the integer remainder of a division
```
i = 27 MOD 5
```

Math

Round Rounds a number to a selectable number of decimal places
```
result = round ( tempvariable,2 )
```
Val Returns the numerical content of a string (**CDbl**)
```
result = Val ("123.4")
```
Int Returns an integer by truncating (different than Fix)
```
i = int ( tempvariable )
```
Fix Returns an integer by truncating (different than Int)
```
i = fix ( tempvariable )
```
Hex Returns the hexadecimal value of any number
```
temp$ = hex ( tempvariable )
```
Oct Returns the octal value of any number
```
temp$ = oct ( tempvariable )
```
Tan Returns the tangent of an angle
```
tempvariable1 = tan ( tempvariable2 )
```
Rnd Returns a random number between 0 and 1
```
tempvariable1 = rnd
```
Randomize Initializes the Rnd function so it gives different answers each time
```
randomize
```
Sgn Returns the sign of a number
```
i = sgn ( tempvariable )
```
Sin Returns the sine of an angle
```
tempvariable1 = sin ( tempvariable2 )
```
Cos Returns the cosine of an angle
```
tempvariable2 = Cos ( tempvariable )
```
Abs Converts a number to a positive value
```
i = Abs ( tempvariable )
```
Sqr Returns the square root of a number
```
tempvariable1 = Sqr ( tempvariable2 )
```
Log Returns the base 10 logarithm of a number
```
tempvariable1 = Log ( tempvariable2 )
```
Atn Returns the arctangent of an angle
```
tempvariable1 = Atn ( tempvariable )
```
Partition Sort of an oddball function but segregates values according to ranges

Type Conversions A variety of conversion functions: `CBool, CByte, CCur, CDate, CDbl, CDec, CInt, CLng, CSng, CStr, CVar`

Loops and Conditional Decisions

If..Then..Else Performs code based on the results of a test
```
If A>5 Then Print "A is a bit number!"
```
For...Next Loops a specified number of times
```
For i = 1 to 5: Print #1, i: next i
```
For Each ... Next Walks through a collection
```
For Each X in Form1.controls: Next X
```
Exit For Exit the FOR loop

While...Wend Loops until an event is false
```
While i < 5: i = i +1: Wend
```
Select Case Takes an action based on a value of a parameter
```
Select Case i
  Case 1 : Print "it was a 1"
  Case 2 : Print "it was a 2"
End Select
```
Do While|Until...Loop Loops until conditions are met
```
Do While i < 5 : i = i + 1 : Loop
Do i = i + 1 Loop Until i = 20
```
Exit Do Exit the DO loop

IIF Returns 1 of two parts, depending on the value of an expression
```
result = IIF (testexpression, truepart, falsepart)
```
Choose Selects and returns a value from a list of arguments
```
Choose (index, "answer1", "answer2", "answer3")
```
With Executes a series of statements on a single object
```
With textbox1
   .Height = 100
   .Width = 500
End With
```
End Immediately stops execution of a program

Stop Pauses execution of a program (can restart without loss of data)

Switch Returns a value associated with the first true expression in a list
```
result=Switch (testvalue1, answer1, testvalue2, answer2)
```
GoTo Switches execution to a new line in the code
```
GoTo Line1
```
GoSub ... Return Switches execution to a new block of code and then returns `GoSub Line1`

On .. GoSub Branch to a specific line of code then return at the next `Return` statement `On Number GoSub Line1, Line2, Line3`

On .. GoTo Branch to a specific line of code
```
On Number GoTo Line1, Line2, Line3
```

Arrays

Arrays usually range from 0 to **UBound**, it means there are **UBound** +1 items in an array. This could be changed by **Option Base**.

Option Base Determines whether the lowest range of a array is 0 or 1

```
Option Base 1
```

Dim Creates an array

```
Dim arrayname(25)
Dim arrayname(5 To 50)
```

ReDim Resets the bounds of an array (**Preserve** saves the values)

```
ReDim arrayname(28)
ReDim Preserve arrayname(28)
```

Erase Erases all values of an array

```
Erase arrayname
```

UBound Returns the upper dimension of an array

```
i = UBound (arrayname)
```

LBound Returns the lower dimension of an array

```
i = LBound (arrayname)
```

Filter Returns a subset of an array based on a filter

```
Filter (inputarray, searchstring)
```

Array It returns an array that has been filled with data from a list. It allows you to put the actual data values in the code to avoid having the user input it or to avoid having to read it from a file

```
ArrayName = Array (10, 20, 30)
```

Join Concatenates strings within an array

Declarations

Dim Used to define a variable as a certain type

```
Dim i As Integer, r As Single
```

Use the **Option Explicit** to make sure that VB forces you to declare every variable you use. **Dim** is the simplest way to declare a variable.

ReDim Used to declare and change the dimensions of a dynamic array

```
Redim arrayname(rabbits)
ReDim Preserve arrayname(rabbits + 47)
```

Static Establishes a procedure variable which keeps its value between calls

```
Static i As Integer
```

For example, if you want to keep track of how many times you've been in a procedure, set a counter as STATIC and increment it by one for each visit to the procedure. It will never go away until the program is terminated.

Public Creates a variable which can be accessed outside its own procedure

```
Public i As Integer
```

Even if you're the only programmer writing code in your application, use of Private vs Public will help catch errors if you inadvertently try to access an out-of-scope variable

Private Creates a variable that can be read only in its own procedure or module, according to where the declaration took place.

```
Private i As Integer
```

Use this as often as possible to avoid unnecessary exposure of your variables to coding mistakes.

Sub Defines a procedure which can execute a block of code

```
Sub newProcedure (var1 As Integer, var2 As String)
```

Be sure to check out HELP for how to handle Sub arguments. There are more questions and mistakes made concerning the use of arguments than just about anything else I've seen.

Function Declares a procedure which can return a value

```
Function newFunction (var1 As Integer, var2 As String)
As Single
```

This is actually the most versatile of the Sub/Function procedure types. It can do anything a Sub can do as well as returning a value for use in an expression.

Call Transfers control to a Sub or Function (is optional, rarely used)

```
Call Procedure 1
```

CallByName Executes a method of an object or set/returns a property

```
CallByName(form1,procedurename,vbMethod)
```

The really cool thing about this is that you don't have to hardcode a procedure call. Just use a string variable with the name of the procedure to call.

Option Explicit Instructs VB to force an explicit declaration of all variables

```
Option Explicit
```

You're borderline stupid if you don't use it to catch typing errors. Set up the VB IDE to automatically include this in all projects.

Option Compare Instructs VB on how to make string comparisons

```
Option Compare Binary
```

This can add case-insensitivity for those times when you don't want to hard-code it

Option Private Prevents a module's content from being referenced outside a project.

```
Option Private Module
```

Generally doesn't apply to most VB applications. If you find a good use for it let me know.

Property Get Declares how to get the value of a property

```
Property Get Name()
```

You won't use this much until you get into creating classes of your own

Property Let Declares how to assign a value to a property

```
Property Let Name()
```

You won't use this much until you get into creating classes of your own

Property Set Declares how to set a variable reference to an object

You won't use this much until you get into creating classes of your own

Set Assigns an object reference to a variable

```
Set X = form1.txtInputFromUser
Set X = ActiveSheet
Set X = Worksheets(„jmeno listu“)
```

Very useful for making code more readable or simply to cut down on how much typing you have to do!

Let Precedes assignment of a value to a variable

```
Let i = 3
```

It's optional, no one uses, so forget you ever saw it

Type...End Type Creates a user defined part type which consists of standard VB data types

```
type anytypename
one as string
two as integer
three as boolean
End Type
```

This is a really excellent way to keep several kinds of data under one variable name. Plus, you can PUT or GET a user-defined type with a single line of code.

Const Creates a variable whose value is fixed

```
o const anyname
```

o Basically, use this to give easy to remember names to values. For example, suppose you use the value 37.2 a lot in your code, then if you put CONST MyAge = 37.2 in your code you'll be able to insert the MyAge where the 37.2 should have gone. Easier to type and easier to read. Also, you can chane the value of the constant by changing only the declaration line of code, rather than searching out every place the value was used!

Declare Used to define a procedure that exists in another file

```
declare functionname (arg1 as integer, arg2 as
   string) as integer
ArrayName = Array (10, 20, 30)
```

Implements Specifies a class to be implemented in a module

Friend Allows procedure to be callable from modules outside the class

GetObject Return a reference to an ActiveX component

CreateObject Creates and returns a reference to an ActiveX object

GetAutoServerSettings Returns information about the state of an **ActiveX** component's registration.

Enum Declares a type for an enumeration

Event Declares a user-defined event

TypeName Returns the type of data in a variable

VarType Returns the type of data in a variable

DefType Sets the default data type of variables

```
DefInt A-Z
```

IS A variety of data type or status checking options

```
IsArray, IsBindable, IsBroken, IsDate, IsDirty,
IsEmpty, IsError, IsMissing, IsNull, IsNumber,
IsObject, IsReady, IsRootFolder
```

Special Values

True A logical (Boolean) expression. In VB, its value is -1

False A logical (Boolean expression. In VB, its value is 0

Nothing Disassociates an object variable from an actual object

Null Indicates that a variable has no valid data

Empty Indicates that a variable has not yet been initialized

Miscellaneous

MsgBox A built-in dialog box that gives a message and allows a user input

```
MsgBox(prompt[, buttons][, title][, helpfile,context])
```

`Chr(13)` – cut the line

`Chr(10)` – feed the line (?)

InputBox - A built-in dialog box that allows entry of a text string

```
InputBox(prompt[,title][,default][,xpos][,ypos][
      ,helpfile,context])
```

DoEvents Allows VB to complete pending tasks

```
doevents
```

Shell Executes a 2nd program from within the current program

```
shell "notepad.exe"
```

Note: VB does not wait for the Shell'd program to quit before executing the next line of code!

Command - Gives any text that followed a VB .EXE execution command

o temp$ = command

Environ - Returns the system environmental space content

o temp$ = environ

Beep - Makes the computer beep once.

o beep

AddressOf - Provides an entry point for an external program to use a procedure

o AddressOf (procedurename)

AppActivate - Activates an applications window
- o AppActivate (windowtitle)

RaiseEvent - Fires an event declared at module level
- o RaiseEvent ProcedureName

Load - Load an object
- o load form1

Unload - Unload an object
- o Unload form1

LoadPicture - Load a picture into a control property
- o form1.picture = loadpicture (filename)

SavePicture - Save a picture to a file
- o SavePicture(form1.picture,filename)

LoadResData - Load the data from a resource file
- o LoadResData(index,format)

LoadResString - Load a string from a resource file
- o LoadResString(index,format)

SendKeys - Send keys to another app as though they were from the keyboard
- o Sendkeys {DOWN}

QBColor - Returns a value corresponding to the original QB values 0-15
- o form1.backcolor = QBcolor (12)

RGB - Returns a color value by inputting the red, green, and blue parts
- o form1.backcolor = RGB (12,128,256)

Me - Refers to the current object, usually the active form
- o print Me.caption

Registry

One thing to remember is that the registry save strings so if you're saving or reading numeric information then may have to do some string manipulation with the results.

GetSetting - Get a value from the Registry
- o temp$ = getsetting "TestApp", "SectionName", "KeyName", "defaultvalue"

GetAllSettings -Returns a list of key settings and their values
- o GetAllSettings(appname,section)

SaveSetting - Save a value into the Registry
- o savesetting "TestApp", SectionName, KeyData

DeleteSetting - Deletes an entry from the registry
- o deletesetting "TestApp", "SectionName", "Keyname"

Error Handling

On Error - Enables an error-handling routine
- o On Error GoTo Line2 (if error occurs, go to line2)
- o On Error Resume Next (if error occurs, continue executing next line of code)
- o On Error Goto 0 (disables error handling)

Resume - Used to resume execution after a error-handling routine is finished
- o Resume
- o Resume Next
- o Resume Line1

CVErr - Returns an error type variable containing a user-specific error number
- o X = CVError(13)

Error - Simulates the occurrence of an error
- o Error 23

File Handling (Generic)

Dir Returns a filename that matches a pattern
```
temp$ = Dir ("*.*")
```
CurDir Returns the current directory
```
temp$ = CurDir
```
MkDir Creates a directory
```
mkdir ( "newdirectoryname" )
```
ChDir Changes the current directory to a new location
```
chdir ( "newdirectoryname" )
```
ChDrive Changes the current drive
```
ChDirve "A"
```
RmDir Removes the indicated directory
```
rmdir ( "directoryname" )
```
Freefile Returns an unused file handle
```
i = freefile
```
Open Opens a file for access, locking it from other applications
```
open "filename" for input as #1
```
Close Closes a file so that other applications may access it
```
close #1
```
LOF Returns the length of a file in bytes
```
i = lof ( #1 )
```
EOF Returns a boolean value to indicate if the end of a file has been reached; `statusvariable = eof ( #1 )`

Name As Renames a file
```
name "filename1" as "filename2"
```
Kill Deletes a file
```
kill "filename"
```
Fileattr Returns attribute information about a file
```
i = int ( tempvariable )
```
GetAttr Returns attributes of a file or directory
```
i = GetAttr("c:\windows\temp")
```
SetAttr Sets the attributes of a file
```
SetAttr pathname, vbHidden
```
Reset Closes all disk files opened by the OPEN statement
```
Reset
```
FileDateTime Returns data file was created or last edited
```
FileDateTime ( filename )
```
FileLen Returns length of file in bytes
```
FileLen ( filename )
```
FileCopy Copies a file to a new name
```
FileCopy sourcefile, destinationfile
```
Lock Controls access to a part or all of a file opened by OPEN
```
Lock #1
```
UnLock Restores access to a part or all of a file opended by OPEN
```
UnLock #1
```
Width # Set the output line width used by the OPEN statement
```
Width #2, 80
```

File Handling - ASCII-specific

Line Input - Reads an entire line of ASCII text
- o line input #1, tempvariable$

Write - Puts data in a file, with separators for the data
- o write #1, tempvariable$

Print - Puts data in a file with no separators
- o print #1, tempvariable$

Spc - Used in a print statement to move a number of spaces
- o Print #2, var1; spc(15); var2

Tab - Used in a print statement to move to TAB locations
- o Print #2, var1; Tab(20); var2

File Handling - Binary-specific

The big difference between the two is that binary access will read (Get) an exact number of bytes of data, and the reading can start at any byte within the file.

Get - Reads data from a file
- o get #1, anyvariable

Put - Puts data into a file
- o put #1, anyvariable

Seek - Moves the current pointer to a defined location in a file
- o seek #1, 26

Input
- o input #1, anyvariable

Loc - Returns current position with an open file
- o i = Loc(#2)

Date/Time

Date - Gets the current date

Time - Gets the current time

Now - Gets the current date and time

Timer - Returns the number of seconds since midnight

DateAdd - Adds a time interval to a date

DateDiff - Returns how many time intervals there are between two dates

DateSerial - Returns the month/day/year

DateValue - Returns the date

Year - Returns the current year

Month - Returns the current month (integer)

MonthName - Returns the text of the name of a month

Day - Returns the current day

Hour - Returns the current hour

Minute - Returns the current minute

Second - Returns the current second

TimeSerial - Returns a date with the hour/minute/second

TimeValue - Returns the time

WeekDay - Returns the current day of the week (integer)

WeekDayName - Returns the text of a day of the week

Financial Calculations

DDB - Returns the depreciation of an asset for a specific time period

FV - Returns the future value of an annuity

IPmt - Returns the interest payment of an investment

IRR - Returns the internal rate of return on a cash flow

MIRR - Returns a modified internal rate of return on a cash flow

NPer - Returns a number of periods for an annuity

NPV - Returns a present value of an investment

`PPmt` - Returns the principal payment of an annuity
`PV` - Returns the present value of an annuity
`Rate` - Returns the interest rate per period for an annuity
`SLN` - Returns the straight-line depreciation of an asset

`SYD` - Returns the sum-of-years' digits depreciation of an asset

Trideni podle abecedy: bubble-sort a quick-sort. Popis a algoritmy najdeš na netu

Set zdroj = Worksheets("lesy")